GeekBrains

Мой первый RecSys проект

Подтема

Предсказание пяти товаров, которые купят покупатели. Датасеты от X5.

Дмитрий Яковлев

Закончил факультет искусственного интеллекта GeekBrains..
Закончил курсы повышения квалификации в МГТУ им. Баумана.

Немного о себе.

- Проживаю в г. Новосибирск, ищу работу в г. Москва.
- Инженер по работе с клиентами и проектировщиками на предприятии, производящем высоковольтное оборудование.

План проекта

1. Пройденный этап: DataPreprocessing
2. Пройденный этап: EDA
3. Пройденный этап: FeatureGeneration
4. Пройденный этап: настройка регрессоров
5. Пройденный этап: создание общего пайплайна
6. Пройденный этап: создание product ready приложения
7. Будущий этап: доработка проекта
9. Будущий этап
10. Будущий этап
11. Будущий этап
12. Будущий этап
13. Будущий этап
14. Будущий этап
15. Будущий этап
16. Будущий этап

Достигнутые цели

```
lgbm_recs.apply(lambda x: precision_at_k(x['lgbm'], x['actual']), axis = 1).mean()*100
```

```
20.384236453201687
```

```
lgbm_recs.apply(lambda x: ap_k(x['lgbm'], x['actual']), axis = 1).mean()*100
```

```
15.100492610837406
```

```
[161] lgbm_recs.apply(lambda x: precision_at_k(x['lgbm'], x['actual']), axis = 1).mean()*100
```

```
86.44028103044474
```

```
[162] lgbm_recs.apply(lambda x: ap_k(x['lgbm'], x['actual']), axis = 1).mean()*100
```

```
85.21350507416065
```

Нашел киллер-фичу user_item_day которая резко увеличила метрики. Значительно увеличило метрики решение отсеять покупателей, купивших менее 84 товаров за исследуемый период. Немного увеличило метрики решение удалить строки, в которых произошла транзакция при нулевых покупках.

В результате всех дополнительных улучшений целевая метрика на тренировочной выборке увеличилась в 5,7 раза по сравнению с вариантом без улучшений.

Но это конечно же не реперезентативный вариант валидации. И с уровнем знаний 2020 года я еще не знал как сделать ее репрезентатаивной.

Решение задачи / План работы

- Исследование рабочего датасета: поиск аномалий, анализ распределений, корреляций, исследование методов работы с количественными и категориальными признаками, исследование выбросов, поиск оптимального алгоритма построения классификатора, определение полезных признаков.
- Генерация новых признаков.
- Разработка классификатора.

Трудности

В датасете юзеры, которые в подавляющем большинстве покупают одно и то же, и перебить baseline, который рекомендует самые популярные товары не так то просто. После добавления киллер-фичи у конечного предсказания дикий рост метрик на валидации, которое почти не зависит от метрики модели на тренировочном датасете.. И дальнейшее увеличение метрики модели трейне приводит к уменьшению метрики на валидации. В тот момент не знал продвинутых методов настройки валидации, поэтому не смог добиться дальнейшего улучшения качества модели.

Это проект 2020 года. В то время продвинутых методов проектирования моделей я еще не знал. Поэтому не судите строго.

Baseline

```
lgbm_recs.apply(lambda x: precision_at_k(x['lgbm'], x['actual']), axis = 1).mean()*100
```

```
20.384236453201687
```

```
lgbm_recs.apply(lambda x: ap_k(x['lgbm'], x['actual']), axis = 1).mean()*100
```

```
15.100492610837406
```

В качестве baseline-решения выбрал простое решение основанное на двухуровневой модели (als + lightgbm). AP@5 = 15,1, precision@5 = 20,38.

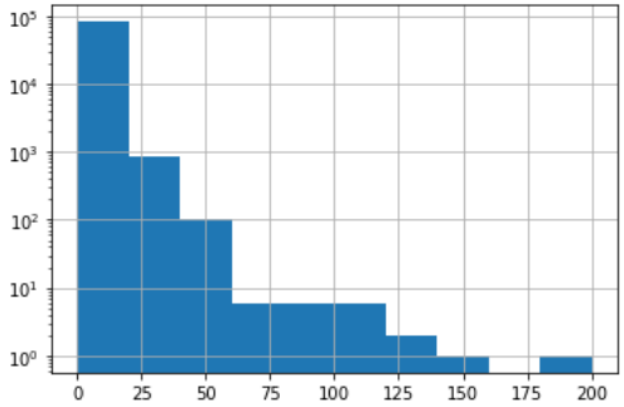
Я понимаю, что по хорошему в качестве базового решения еще надо было бы рассмотреть вариант с 5 самыми популярными товарами, но сделаю этот вариант чуть позже.

EDA

sales_value

```
data_train.loc[data_train['sales_value'] < 10000, 'sales_value'].hist()

plt.yscale('log')
```



Часть количественных признаков имеют логнормальное распределение с выбросами. Поскольку это задача классификации с «деревянным» классификатором на выходе, смысла приводить их к нормальным распределениям отсутствует. Проблема выбросов в задачах классификации тоже второстепенна. Более важно найти аномалии. Часть аномалий была найдена, но думаю что надо еще покопаться в данных.

FeatureGeneration

[91]:
```
user_item_day = data_train.groupby(['user_id', 'item_id'], as_index=False).agg({'day':'max'})\ax'})\
                          .rename(columns={'day':'user_item_day_max'})
user_item_day.head()
```

:[91]:

	user_id	item_id	user_item_day_max
0	1	819312	536
1	1	820165	610
2	1	821815	311
3	1	823721	291
4	1	823990	146

[92]:
```
targets_lvl_2 = targets_lvl_2.merge(user_item_day, on=['user_id', 'item_id'], how='left')
```

Киллер-фичой, перетянувшей на себя всю важность в классификаторе оказался признак user_item_day. По неопытности никак не обработал эту ситуацию и поэтому другие сгенерированные признаки оказались бесполезными.

Выбор класификатора

```
%%time
from sklearn.model_selection import RandomizedSearchCV
n_estimators = [int(x) for x in np.linspace(start = 10, stop = 1000, num = 10)]
max_features = ['log2', 'sqrt']
max_depth = [int(x) for x in np.linspace(start = 1, stop = 15, num = 10)]
reg_lambda = [int(x) for x in np.linspace(start = 0.1, stop = 1, num = 5)]
reg_alpha = [int(x) for x in np.linspace(start = 0.1, stop = 1, num = 5)]

param_dist = {'n_estimators': n_estimators,
              'max_features': max_features,
              'max_depth': max_depth,
              'reg_lambda': reg_lambda,
              'reg_alpha': reg_alpha
               }

rs = RandomizedSearchCV(lgb,
                        param_dist,
                        n_iter = 100,
                        cv = 3,
                        verbose = 1,
                        n_jobs=-1,
```

В этом проекте выбирал между LGBMClassifier и LGBMRanker. LGBMRanker показал на 0,1% результат выше, выбрал его. В то время не знал, что в подобных случаях классификатор надо выбирать с помощью a/b-теста.

Гиперпараметры выбирал с помощью RandomizedSearchCV и подобрать не смог. Продвинутые методы подбора гиперпараметров и методов валидации в 2020 году еще не знал.

Предложения по проекту

В первую очередь более качественно проработать EDA, поискать скрытые аномалии, зависимости. Качественно настроить валидацию. Обработать признак user_item_day так, чтобы он не перетягивал на себя всю значимость в классификаторе. Подобрать другие признаки, производные от существующих. Произвести настройку гиперпараметров более продвинутыми методами, нежели RandomizedSearchCV . Произвести выбор нескольких классификаторов, в т.ч. несколько нейронных сетей и попытаться настроить ансамбль классификаторов.